Hybrid reasoning for multi-robot drill planning in open-pit mines

Mansouri, Masoumeh; Andreasson, Henrik; Pecora, Federico

DOI:
10.14311/APP.2016.56.0047

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (Harvard):
Mansouri, M, Andreasson, H & Pecora, F 2016, 'Hybrid reasoning for multi-robot drill planning in open-pit mines', *Acta Polytechnica*, vol. 56, no. 1, pp. 47-56. https://doi.org/10.14311/APP.2016.56.0047

Acta Polytechnica **56**(1):47–56, 2016
doi:10.14311/APP.2016.56.0047

available online at http://ojs.cvut.cz/ojs/index.php/ap

HYBRID REASONING FOR MULTI-ROBOT DRILL PLANNING IN OPEN-PIT MINES

Masoumeh Mansouri*, Henrik Andreasson, Federico Pecora

Centre for Applied Autonomous Sensor Systems (AASS)
* corresponding author: masoumeh.mansouri@oru.se

Abstract. Fleet automation often involves solving several strongly correlated sub-problems, including task allocation, motion planning, and coordination. Solutions need to account for very specific, domain-dependent constraints. In addition, several aspects of the overall fleet management problem become known only online. We propose a method for solving the fleet-management problem grounded on a heuristically-guided search in the space of mutually feasible solutions to sub-problems. We focus on a mining application which requires online contingency handling and accommodating many domain-specific constraints. As contingencies occur, efficient reasoning is performed to adjust the plan online for the entire fleet.

Keywords: robot planning; multi-robot coordination; on-line reasoning.

1. Introduction

Autonomous vehicles are becoming key components in industrial automation. State-of-the-art methods for localization, mapping, control and motion planning have enabled the development and deployment of autonomous vehicles in logistics, material handling, and mining application. Yet important hindrances remain when it comes to employing fleets of autonomous vehicles [1]. This paper is concerned with three of these hindrances:

- Fleet automation often involves solving several strongly correlated sub-problems — among these, allocating tasks to vehicles, planning vehicle motions, and vehicle coordination. This makes solving the overall problem very hard, as solutions should be found in the space of mutually-feasible solutions to sub-problems.
- Solutions often need to account for very specific, domain-dependent constraints. Thus, even if methods exist for solving individual sub-problems, these often need to be adapted to reflect the nuances of the particular domain.
- Several aspects of the overall fleet management problem become known only online. This makes it necessary to compute at least parts of the solution to the overall problem during execution and/or to adapt existing plans to contingencies.

In this paper, we propose a method for dividing the overall fleet-management problem into sub-problems. We apply a general method for searching for a mutually feasible solution to the sub-problems of the overall problem. We focus on a mining application where fleets are composed of surface drilling machines. The aim is to plan and coordinate blast-hole pattern drilling with multiple drilling machines. Solutions consist of an executable plan for multiple vehicles operating concurrently within a common area of the open-pit mine, called a *bench* (see Figure 1). Several aspects of the problem become known only online: the duration of hole-drilling actions depends on the hardness of the rock, which is unknown at planning time; the durations of navigation actions between targets are also unknown at planning time, as they depend on the actual state of the terrain; and unplanned stops may occur due to the fact that drills may get stuck and need to be replaced. All three forms of contingencies are modeled as metric temporal constraints, and are posted online to a common representation that maintains the state of execution of the plan for the entire fleet. As temporal constraints become known during execution, efficient temporal reasoning is performed to adjust the plan and to provide an optimistic Time to Completion (TTC) of the overall plan for the entire fleet.

Figure 1. Two AtlasCopco drilling machines (Pitviper-351) in the process of drilling targets in a bench.

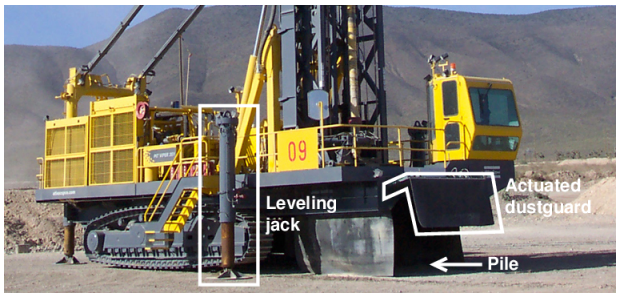

Figure 2. The actuated dust guard and leveling jacks are highlighted on an Atlas Copco Pitviper-271. After drilling, piles of excess material accumulate under the dust guard, which requires the machine to actuate the dust guard before navigating away from the target.

In Section 2 we define the Drill Pattern Planning Problem and state the specific requirements of the application. Section 3 details how the problem is divided into sub-problems, and Section 4 describes the algorithm used to find mutually feasible solutions to the individual sub-problems. Section 5 describes the domain-specific heuristics that capture the nuances of this particular mining application and how they can be easily plugged into the search for a solution. Section 6 is dedicated to the online aspects of the approach. A preliminary experimental evaluation is provided in Section 7, which evaluates the feasibility of on-line plan adaptation in a simulated environment. A brief discussion on related work and on the outlook concludes the paper.

2. Problem Definition and Requirements

We focus on a problem in a mining application, where a fleet of surface drills operates on a bench in an open-pit mine. A set of *drill targets* in the bench is given; at each target, a blast hole is to be drilled. The blast holes are then filled with explosive material that will be detonated after all targets have been drilled. After the explosion, the ore is taken away and processed for mineral extraction.

For each drill target, a machine can autonomously carry out a set of tasks: auto-tramming (navigating to the target from its current position), leveling (deploying jacks for horizontally leveling the machine), drilling, and de-leveling (retracting the jacks so the machine is placed back on its tracks). Each drilling machine has a square dust guard around its drill bit. Drilling produces piles of excess material around the hole, thus one side of the dust guard must be lifted after drilling; this allows the machine to navigate to the next target without colliding with the pile that has accumulated under it after drilling (see Figure 2).

The *Drill Pattern Planning Problem* (DP^3) consists of computing a plan that involves machines reaching each drill target in a bench and performing the necessary operations to drill the blast hole. The plan is subject to the following requirements:

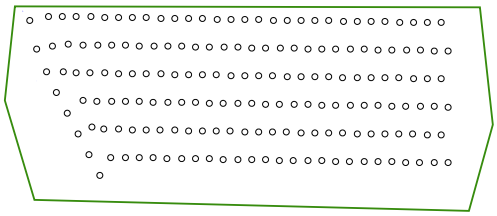

Figure 3. A bench with drill targets (grey circles) and a geofence (green polygon).

- Machines should not collide with obstacles/each other;
- Drilling a target leads to the creation of a pile under the dust guard of the machine; this pile constitutes an obstacle, hence no machine can drive over it;
- Once drilling has been performed, the machine can only drive away from the target (backing up from the target pose) by raising the forward dust guard (the only part of the dust guard that can be actuated);
- Motions should be executable by the machines i.e., motions should be kinematically feasible;
- It should be possible to modify the plan online;
- The plan should respect spatial constraints such as a virtual fence within which the vehicles are allowed to operate; this area is called a *geofence*.

The locations of the drill targets and the geofence are given. These are based on a geological survey of the area and on the current production targets of the mine (see Figure 3). A set of machines $\mathcal{R}$ is also given, and the size of a fleet is based on the size of the bench. Also given are initial poses of all machines, as well as their desired final "parking" poses.

The requirements above pose several problems e.g., task allocation (of machines to target poses), motion planning, and coordination. These problems cannot be treated separately, as the solutions of each problem depend on each other. For instance, coordination must lead to a sequence of target poses that accounts for the piles generated after drilling (which become obstacles that must be taken into account in motion planning). Hence, it is necessary to subject the possible choices made to solve one problem to the choices made in resolving the other problems — e.g., verifying through motion planning that a chosen sequence of targets to drill will be kinematically feasible. Because of these interdependencies, we face a hybrid reasoning problem. We propose an approach in which the overall problem is divided into sub-problems, and the solution to the overall problem is searched for in the joint search space of these sub-problems. In the next section, we define each sub-problem in detail, while in Section 4 we outline the algorithm used to search for plans that are mutually feasible with respect to all the sub-problems.

3. Approach

We divide DP^3 into five sub-problems.

- The *sequencing sub-problem* consists of deciding a total ordering of targets i.e., sequencing every pair of targets.
- The *motion planning sub-problem* consists of deciding the pose of the drilling machines at each target. Constraints on the orientation of the machine in certain target poses may be given (e.g., due to the presence of the geofence or other geographical constraints like cliffs and walls). The decisions are subject to kinematic constraints, obstacles and the geofence, and must account for piles resulting from drilling, as well as the dust guard mechanism.
- The *machine allocation sub-problem* consists of allocating machines to targets given the available machines and their positions. Machine allocation also accounts for the need to reach a given end parking position.
- The *coordination sub-problem* consists of scheduling machines. Solutions to this sub-problem consider the spatio-temporal overlap between machines and between machines and piles.
- The *temporal sub-problem* consists of deciding when machines should carry out motion, drilling, leveling and de-leveling operations, subject to temporal constraints arising from coordination, sequencing, maximum achievable speeds, etc.

A solution to DP^3 is obtained by reasoning upon these five different sub-problems jointly. Candidate solutions for a sub-problem are *validated* by dedicated solvers. Each solver focuses on a subset of aspects of the overall problem, e.g., a motion planner verifies kinematic feasibility and absence of collisions, while a temporal solver verifies that coordination choices are temporally feasible. Validated solutions for each sub-problem can be seen as *constraints* that account for particular aspects of the overall problem. They are maintained in a common representation, which is sufficiently expressive to model the search space of all problems jointly. The common representation is a constraint network where variables represent missions. A *mission* is a tuple $M = (gp, sp, r, P, m, S, T, A)$, where r represents the robot which should perform the set of activities $A = \{\text{drilling}, \text{leveling}, \text{de-leveling}\}$ at, respectively, starting pose sp and goal pose gp. P is the path that r traverses to reach gp from sp, and is computed based on a map m of the environment. S is a set of polygons representing sweeps of the robot's footprint over P, and T is a set of time intervals representing when r will be in each polygon contained in S. Henceforth, we denote with $M(\cdot)$ an element of the mission tuple.

Let $\mathcal{M}$ be the set of all missions in DP^3 (one for each drill target). A solution to DP^3 is such that a value is decided for all elements of a mission, for each mission in $M \in \mathcal{M}$. Each element is decided by solving one or more sub-problems. We view a mission M as a variable in a Constraint Satisfaction Problem (CSP, see [2]) whose domain represents all possible combinations of values that can be given to each element of M. Accordingly, we view a sub-problem as the problem of constraining the domains of missions so that the requirements stated above are met. Hence, the solvers that validate solutions to the sub-problems are seen as procedures that post constraints to the common constraint network. As we will see, adopting the CSP metaphor allows us to employ heuristic search strategies for solving the overall DP^3.

3.1. Sequencing Sub-problem

The sequencing sub-problem consists of finding a total order of missions. A decision variable of this sub-problem is a mission $M_i \in \mathcal{M}$ that does not have a preceding mission. A possible value that can be assigned to this decision variable is a precedence constraint $M_j \xrightarrow{\text{precedes}} M_i$, asserting that mission $M_j \in \mathcal{M}$ should occur before M_i. M_j is a mission for which it has not been already decided that it precedes another mission. A sequencing solver verifies that missions are totally ordered. Figure 4(a) shows an example of decision in this sub-problem, namely $M_{166} \xrightarrow{\text{precedes}} M_{137}$. White arrows in the figure represent precedence constraints.

3.2. Motion Planning Sub-Problem

The motion planning sub-problem consists of finding a goal pose gp for each mission $M_i \in \mathcal{M}$. A gp is a tuple $\langle x, y, \theta \rangle$ in which x and y represent the position of a drill target, and θ is the orientation of the machine. The decision variables of the motion planning sub-problem are missions M_i such that (1) $M_i(gp)$ does not have a defined orientation, i.e., θ is not assigned to an angle, (2) there exists $M_j \xrightarrow{\text{precedes}} M_i$ in the common constraint network, and (3) $M_j(gp)$ has been assigned an orientation. Possible values that can be assigned to a decision variable are a set of eight angles $\{\theta_1, \ldots, \theta_8\} \in [0, 2\pi)$.

A particular choice of approach angle for a mission is only feasible if the machine can drive away from the previous target $M_j(gp)$ and can navigate to the end pose of a mission $M_i(gp)$ considering piles created by all the preceding missions. For example, Figure 4(b) shows a selection of one feasible approach angle for several missions with respect to existing sequencing constraints. The approach angles are represented by pink arrows, and the machines drive away from the targets in the opposite direction of the pink arrows.

The eight possible assignments to the decision variable determine eight different possible end poses of the machine $\{M_i(gp_1), \ldots, M_i(gp_8)\}$, which differ only in the orientation of the machine in the goal pose. A possible assignment θ_k is validated

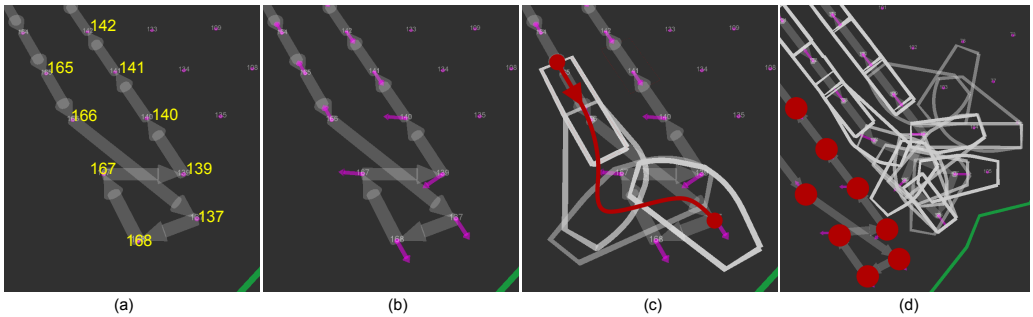

FIGURE 4. Examples of different sub-problems with their decision variables and particular choices of values.

through a path planner, which is given the triple $(M_i(sp), M_i(gp_k), M_i(m))$, where the start pose is the goal pose of the preceding mission $(M_i(sp) = M_j(gp))$, and $M_i(m)$ is a map of the environment that contains obstacles and a geofence. The obstacles correspond to circular shapes centered in the goal poses of the preceding missions. Through the map $M_i(m)$, the path planner accounts for targets that have already been drilled prior to mission M_i.

Since the path planner is potentially invoked several times while solving the motion planning sub-problem, computational efficiency is crucial. We employ a path planner for a car-like mobile robot based on cubic spirals [3]. This path planner computes paths consisting of curvature-constrained curves constituted by a few cubic spirals and straight lines. The output of the path planner is either `fail`, which indicates that a particular approach angle θ_k cannot be achieved, or the spline $M_i(P)$, representing a kinematically-feasible and obstacle-free motion from $M_i(sp)$ to $M_i(gp_k)$.

3.3. MACHINE ALLOCATION SUB-PROBLEM

In this sub-problem, a decision variable is a set $\mathcal{M}' \subseteq \mathcal{M}$ such that $\forall M \in \mathcal{M}'$, $M(r)$ has not been decided and

$$\exists M_i \in \mathcal{M}' : M_i \xrightarrow{\text{precedes}} M \vee M \xrightarrow{\text{precedes}} M_i,$$

that is, a total order of missions has been decided, but machines have not been allocated. The values are complete assignments of robots to missions, i.e., an assignment $M(r) = R$ for each mission in $\mathcal{M}'$.

Clearly, the machine allocation sub-problem has a huge space of possible solutions. Each possible solution has complex ramifications on other sub-problems: different allocations will affect the amount of coordination necessary; allocations must be such that the final mission of a machine is not surrounded by piles (drilled by other machines), which would make it impossible to navigate to its final parking pose. As we show in Section 5, heuristics with high pruning power are needed to explore the search space of this sub-problem, and these heuristics must account for other sub-problems. Solutions to the machine allocation sub-problem are indirectly validated in other problems, hence no particular solver is used for direct validation of possible values.

3.4. TEMPORAL SUB-PROBLEM

A mission's path P is segmented into a sequence of sub-paths based on its curvature. Each segment is associated to a convex polygon s_k resulting from the sweep of the machine's footprint along the sub-path. The resulting sequence $\{s_1, \ldots, s_m\}$ of convex polygons represents the areas occupied by a robot while navigating along the path (see an example in Figure 4(c)). Since the path planner used to obtain P is aware of the obstacles created by preceding missions, the convex polygons do not intersect the piles resulting from drilling (see Figure 4(d), where red circles represent piles, white polygons represents the motions of the machine, and the green line represents the geofence).

In addition to the polygons representing the motion of machines, the activities involved in a mission M (i.e., $M(A) = \{\text{drilling}, \text{leveling}, \text{de-leveling}\}$) have polygons associated to them. Since these activities all occur while the machine is idle in pose $M(gp)$, their polygons coincide with the polygon that covers the last sub-path of $M(P)$. Hence, the set of all convex polygons of mission M is $M(S) = \{s_1, \ldots, s_m\} \cup \{s_{\text{drilling}}, s_{\text{leveling}}, s_{\text{de-leveling}}\}$, where $s_{\text{drilling}} = s_{\text{leveling}} = s_{\text{de-leveling}} = s_m$.

Each convex polygon in $M(S)$ is associated to a time interval in the set $M(T) = \{I_1, \ldots, I_{m+3}\}$. The interval $I_k = [I_s, I_e]$ is a flexible temporal interval within which robot $M(r)$ is in s_k, where $I_s = [l_s, u_s], I_e = [l_e, u_e], l_{s/e}, u_{s/e} \in \mathbb{N}$ represent, respectively, an interval of admissibility of the start and end times of the occurrence of polygon $s_k \in S$.

The temporal sub-problem consists of deciding a start and end time for each interval I_k. The temporal sub-problem has a decision variable for every machine $R \in \mathcal{R}$. Each decision variable is a set of missions $\mathcal{M}' \subseteq \mathcal{M}$ such that for all $M_i \in \mathcal{M}'$, (1) $M_i(P)$ has been decided, (2) $M_i(r) = R$, and (3) the start and end times of $M_i(T)$ have not been decided. We reduce the problem of deciding valid start/end times of the intervals to a Simple Temporal Problem (STP, [4]). The STP is formulated as a collection of temporal constraints as follows. First, for each $M_i \in \mathcal{M}'$ with intervals $M_i(T) = \{I_1, \ldots, I_{m+3}\}$, temporal constraints that reflect the order of the convex polygons

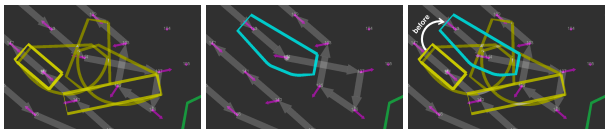

FIGURE 5. Coordinating two machines to avoid spatio-temporal overlap.

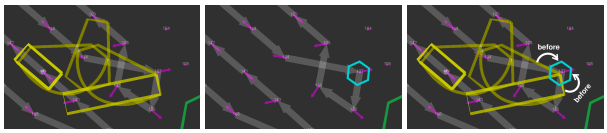

FIGURE 6. Scheduling a machine to avoid spatio-temporal overlap with a pile.

along the path $M_i(P)$ are imposed:

$$I_{k-1} \xrightarrow{\text{before}} I_k,\ k \in \{2 \dots m+3\}. \tag{1}$$

Second, temporal constraints that force the possible start and end times of missions to adhere to the ordering decided by the sequencing solver are added. That is, for each pair of missions $(M_i, M_j) \in \mathcal{M}' \times \mathcal{M}'$ that are subject to a sequencing constraint $M_i \xrightarrow{\text{precedes}} M_j$, the following temporal constraint is added among the intervals $M_i(T) = \{I_1^i, \dots, I_{m+3}^i\}$ and $M_j(T) = \{I_1^j, \dots, I_{m+3}^j\}$:

$$I_{m+3}^i \xrightarrow{\text{before}} I_1^j, \tag{2}$$

reflecting the fact that the de-leveling polygon of mission M_i occurs before the first motion of mission M_j. Third, we impose minimum durations of the motions and activities of the machines:

$$\text{Duration}[\alpha, \infty) \colon I_k,\ k \in \{1 \dots m+3\}. \tag{3}$$

For every motion polygon $s_k, k \in \{1, \dots, m\}$, an initial value for α is computed based on the maximum allowed velocity of machine R. Hence, the earliest time solution of the STP represents the fastest possible execution of all motions and activities in the plan, i.e., an "optimistic" estimate of the start and end times of all operations of all machines. During execution, further temporal constraints can be added to reflect contingencies such as machine maintenance, delays, and so on. The association of an interval per polygon allows us to predicate via temporal constraints on how long every movement or activity will take. Solving the STP is polynomial in the number of intervals in the temporal problem, namely $\sum_{M_i \in \mathcal{M}'} |M_i(T)|$ [4].

Note that there are no temporal constraints among intervals pertaining to different machines, hence the motions and activities of different machines may be concurrent.

3.5. Coordination sub-problem

Since several machines operate in the same environment, it is crucial to address collision/deadlock avoidance. As a consequence of decisions made in all previous sub-problems, the common constraint network includes polygons, temporal intervals, and temporal constraints (eqs. (1) to (3)) among them. The STP solver computes start and end times for each interval. This determines when machines will occupy motion and activity polygons in the various missions. If two polygons pertaining to different vehicles overlap, and their corresponding temporal intervals intersect, then the two vehicles may collide. Coordination avoids this by imposing additional constraints that eliminate temporal intersection where needed. Decision variables of the coordination sub-problem are pairs of polygons and intervals represented by the quadruple $(s_k^i, s_m^j, I_k^i, I_m^j)$, of mission i and j respectively, that overlap both spatially and temporally, i.e., $s_k^i \cap s_m^j \neq \emptyset \wedge I_k^i \cap I_m^j \neq \emptyset \wedge M_i(r) \neq M_j(r)$. The value of a decision variable is one of two possible constraints $\{I_k^i \xrightarrow{\text{before}} I_m^j,\ I_m^j \xrightarrow{\text{before}} I_k^i\}$, imposing either of which eliminates the temporal overlap between concurrent polygons. The STP solver will validate the sequencing in time of these two overlapping polygons accordingly. It will also compute the consequent shift in the occurrence of any other polygon whose interval is constrained with I_m^j or I_k^i by means of temporal constraint propagation within the common constraint network. We use a similar approach for the coordination of multiple vehicles as described in [1]. Figure 5 depicts the situation where missions of two machines are temporally and spatially overlapping. Polygons with the same color belong to one machine and for clarity, we omit the time intervals of polygons in the visualization.

The polygons involved in the decision variables represent two types of occupancy. The first type corresponds to the motions of machines as described in 3.4. The second corresponds to the piles created by drilling. By modeling both types of polygons in the common constraint network, collisions among machines and with piles are found and thus scheduled. Figure 6 illustrates a conflicting situation between the motions of a machine (depicted as yellow polygons) and a pile (depicted as a blue polygon).

4. Backtracking Search

The collection of decision variables for each sub-problem mentioned above constitutes a high-level CSP (henceforth called a meta-CSP). A search in the meta-CSP consists in finding an assignment of values to decision variables that represent high-level requirements. Each of these requirements is, in our case, a sub-problem. Possible values among which these assignments are selected are verified by a specific solver for each sub-problem. Thanks to the common representation of the search space, each sub-problem solver accounts for the assignments made for decision variables of other sub-problems. For example, the path planner validates with respect to a map containing

```
Function DP3-solver(M): success or failure
1  D_seq ∪ D_alloc ∪ D_time ∪ D_coord ∪ D_mp ← CollectDVars(M)
2  if ∃D_i ≠ ∅, i ∈ {seq, alloc, time, coord, mp} then
3      p ← Choose({seq, alloc, time, coord, mp}, h_prob)
4      d ← Choose(D_p, h_i^var)
5      V_d ← CollectValues(d)
6      while V_d ≠ ∅ do
7          v ← Choose(V_d, h_i^val)
8          Update(M, v)
9          if Solve-p(M) then
10             return DP3-solver(M)
11         Remove(M, v)
12         V_d ← V_d \ v
13     return failure
14 return success
```

FIGURE 7. Algorithm DP^3-solver.

obstacles resulting from sequencing decisions; and the coordinator's decisions depend on the machine allocation as well as motion plans. The choices of values for decision variables in the various sub-problems contribute parts of the missions in the common representation, and the sub-problem solvers propagate the consequence of these decisions.

The sub-problem solvers used in our approach are denoted in the following with solve-p, where $p \in \{\texttt{seq}, \texttt{alloc}, \texttt{time}, \texttt{coord}, \texttt{mp}\}$. As we have explained, solve-seq disallows sequencing decisions that are not totally ordered; solve-mp verifies by means of a motion planner that motions are kinematically feasible and obstacle-free; solve-alloc accepts all candidate allocations, as the infeasible ones are discovered indirectly via coordination; solve-time is an STP solver which computes feasible start/end times of mission intervals subject to temporal constraints; solve-coord is also provided by the same STP solver, which validates and computes the consequences of temporal ordering decisions.

We use a CSP-style heuristically guided backtracking search to find values to assign to the decision variables. Henceforth, let the set of sub-problems be indicated by the symbols $\{\text{seq}, \text{alloc}, \text{time}, \text{coord}, \text{mp}\}$. Given the set of missions $\mathcal{M}$, Algorithm DP^3-solver collects all the decision variables belonging to all the sub-problems (line 1), and terminates when no decision variables are left (lines 2 and 14). A particular sub-problem is then chosen according to a sub-problem ranking heuristic h_{prob} (line 3), e.g., h_{prob} prioritizes machine allocation decision variables over coordination decision variables, as the latter problem requires machines to be assigned to missions (see Section 3.5). Among the decision variables of a sub-problem, one is chosen according to a variable ordering heuristic h_i^{var} (line 4). For example, which target should be selected first among the decision variables D_{mp} of the motion planning sub-problem. Among possible alternative values, one is chosen according to a value ordering heuristic h_i^{val} (lines 5–7). For instance, which approach angle has to be selected for a given target in the motion planning sub-problem. This value is added to the common representation (line 8). The sub-problem solver solve-p verifies that assignment v is feasible. If so, DP^3-solver is called recursively (line 10), which results in selecting another unassigned variable subject to the newly updated common representation $\mathcal{M}$. Note that if all possible values are attempted for a decision variable d and all are rejected by solve-p, the algorithm returns a failure (lines 6, 11–13). In the next section, we will explain the problem-, variable- and value-ordering heuristics that are used in DP^3-solver.

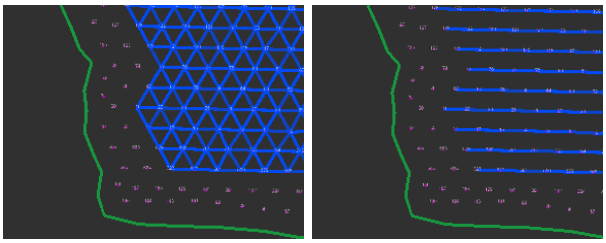

FIGURE 8. An example of topology and group extraction.

5. HEURISTICS

DP^3-solver must select a set of decision variables pertaining to a sub-problem from the union of all decision variables. This selection is guided by a heuristic h_{prob}. Let $D_i \prec D_j$ indicate that the decision variables of problem i have a higher priority than those of problem j. The partial ordering based on which the h_{prob} heuristic operates is $\{D_{\text{seq}} \prec D_{\text{aloc}} \prec D_{\text{tp}} \prec D_{\text{coord}}, D_{\text{mp}} \prec D_{\text{aloc}} \prec D_{\text{tp}} \prec D_{\text{coord}}\}$. Decision variables to branch on (within a chosen D_i) are ordered based on h_{var}, and alternative values are chosen according to h_{val}.

Variable ordering heuristics are provided for the sequencing sub-problem and for the coordination sub-problem. The latter heuristic is based on temporal flexibility, and has been used for resource-constrained project scheduling [5]. The former is based on an analysis of the drill target placements, and is described below.

Variable Ordering for Sequencing $h_{\text{seq}}^{\text{var}}$. The pattern of drill targets is analysed to reveal its topology and the possible principal directions of drill target sequencing (see Figure 8). To determine the former, we use a distance threshold; the latter are discovered via K-Means clustering of the set of angular coefficients of topologically neighbouring drill targets. This yields clusters containing similarly oriented edges of the topology. These are used to group drill targets into roughly-parallel lines (see Figure 8). The topology and the groupings are used to rank drill targets in groups. Variability in these groups are first in the sequencing sub-problems.

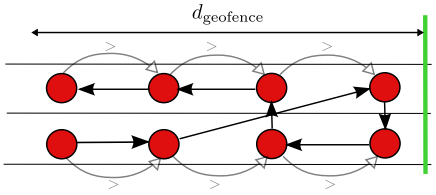

FIGURE 9. An example of a sequence pattern given by an operator.

Value ordering heuristics are defined for the sequencing, allocation, motion planning, and coordination sub-problems. As for variable ordering, the decision variable in the coordination sub-problem are branched upon using a heuristic based on temporal flexibility that is widely used in the scheduling literature [1]. The remaining heuristics $h^{\text{val}}_{\text{seq}}$, $h^{\text{val}}_{\text{alloc}}$are explained below.

Value Ordering for Sequencing $h^{\text{val}}_{\text{seq}}$. A value for a decision variable of the sequencing sub-problem decides which drill target precedes a given target. There are many alternatives for this choice. Note that the sequencing decision variables that are resolved first are those pertaining to drill targets along groups — for such decision variables, the heuristic prioritizes one of two possible predecessors, namely those adjacent to the current decision variable in the grouping. For example, the two values with the highest heuristic score for decision variable M_{141} in Figure 4(a) are $M_{142} \xrightarrow{\text{precedes}} M_{141}$ and $M_{140} \xrightarrow{\text{precedes}} M_{141}$.

Also, this heuristic contributes to alleviating the computational burden of finding sequences in regions close to the geofence while transitioning between groups. Finding a feasible sequence in these situations is challenging, because a machine has limited space to manoeuvre. These regions of highly-constrained motion typically span eight targets for every pair of adjacent groups, thus in the worst case sequencing requires verifying, through motion planning, 8^7 possible motions for each pair. For this reason, the heuristic uses given sequence patterns that reflect common practice by human operators. A sequence pattern is a topological description of a human driving behavior, augmented with metric information that facilitates assessing whether the pattern is applicable in a given region. Specifically, a sequence pattern is a graph (V, E) where V is a set of nodes representing drill targets, and E is a sequence of precedence constraints among the nodes. A distance threshold d_{geofence} is also given, and represents the minimum distance to the geofence required for the pattern to hold. Also, a ranking $<$ of the nodes in terms of how far they lie from the geofence is provided. An example pattern is shown in Figure 9. If the search is considering a decision variable M_i that is surrounded by targets that can be mapped to the nodes in the pattern, then the heuristic ranks possible predecessors of M_i according to the edges in E.

Value Ordering for Motion Planning $h^{\text{val}}_{\text{mp}}$. This heuristic suggests approach angles similar to those assigned to other drill targets in a same group.

Value Ordering for Machine Allocation $h^{\text{val}}_{\text{alloc}}$. A solution to the machine allocation sub-problem determines which drill target is drilled by which machine. Among all possible choices, those are preferred which have three properties: (1) each machine is assigned to a contiguous sequence; (2) the start and end missions of a contiguous sequences are the drill targets close to an open area, i.e., not close to a geofence and not those that are entirely surrounded by other drill targets; and (3) targets are evenly distributed among machines. This heuristic not only contributes to the plan quality in terms of similarity to what a human planner would decide, but also improves the efficiency in planning time by suggesting a restriction on start and exit points for each machine.

6. Adapting Solutions Online

Several aspects of DP^3 are unknown at planning time. For example, the actual durations of activities only become apparent during execution. In a bench, various types of contingencies may occur, such as unexpected maintenance of machines, or increased drilling time due to unknown geological characteristics of the terrain. Therefore, we need to monitor the execution and reflect the contingencies in the common representation. In our approach, the nominal behavior of the machines is given by a solution of the DP^3, obtained via Algorithm `DP3-solver`. The start and end times associated to the intervals $M(T)$ of every mission M are computed through temporal propagation. All the lower bounds represent the earliest possible times at which missions can be executed, and are used to compute the desired speeds at which the computed paths should be driven by the vehicle executives. A machine executive realizes the interface between machine controllers and the missions in the plan by instructing the machine controller to follow the given trajectories[1]. It also updates the time intervals $M(T)$ of the current mission by posting into the common representation constraints representing the current progress of the machine. These constraints are used by the STP solver to propagate any mismatch between prescribed and executed missions of all machines in the fleet.

The STP solver plays a central role in execution monitoring. Machine executives update the common representation at a frequency of 1Hz to dispatch or end missions. A mission is ended by adding a temporal constraint into the common constraint network representing the finish time of the mission as the executive layer informs. The consequences of such updates can be easily computed within the period of one second because the STP solver performs polynomial inference.

[1] In the current implementation, we employ a Model Predictive Controller (MPC) [6].

Also, due to the fact that adding constraints cannot "undo" other decisions, we can post unforeseen durations (e.g., encountering hard rock while drilling) at execution time. More precisely, the prolongation of an activity represented by a flexible time interval associated to a motion polygon cannot affect the sequencing, the approach angle, the particular motions, nor machine allocations. It only bears consequences on the coordination sub-problem, as delays may need to be propagated to other waiting machines. An example of this situation is described in the next Section. Note that if we want to minimize the TTC, then, prolongation of an activity should lead to re-allocation of the machines, which in turn would result in updating the decisions in the sequencing sub-problems. This allows for re-balancing the workload among the machines. Considering the wide range of re-planning strategies that can be used to minimize TTC is a topic for future work.

On-line temporal reasoning also caters to another important requirement of mining companies, namely the need to know an estimate of the Total Time to Completion (TTC). At planning time, we provide an optimistic TTC by initializing the duration constraints 3 with reasonable values: the durations of intervals corresponding to motion polygons are computed using the maximum allowed speed of the machines; and the intervals corresponding to activity polygons are initialized with durations under nominal conditions (average rock density, and no maintenance). As execution proceeds, TTC is updated as a result of temporal reasoning to reflect the actual situation.

7. Experiments and Evaluations

We carried out several experiments in different selections of benches and drill targets. In this section, we exemplify one of the most challenging scenario. The resulting plan was then run by a Gazebo-simulated fleet of Pitviper-311s, and all interfaces between the `DP`^{3}`-solver` and the platforms were realized as ROS [7] nodes. The `DP`^{3}`-solver` ran on a 3.40GHz×4 Intel i7-3770M CPU with 8GB of memory.

The difficulty of the example originates from the closeness of some targets to the geofence. It is also affected by the average distance between targets, which is 16 meters (only 1.8 meters greather than the length of the Pitviper-311). The drill target positions are taken from a real blast hole pattern recently drilled in an open-pit iron-ore mine in Western Australia. As noted earlier, problems become much more difficult if they contain drill targets that are close to the geofence. When that is not the case the problems are easier, since machines have enough space to manoeuvre regardless of how the approach angles are chosen. This experiment contains 76 drill targets and 3 available machines. Figure 10 shows the final solution to the overall DP^3 for this specific bench. In the figure, robots and groups used by $h^{\text{var}}_{\text{seq}}$ and $h^{\text{val}}_{\text{seq}}$ are numbered to facilitate the explanation. As shown,

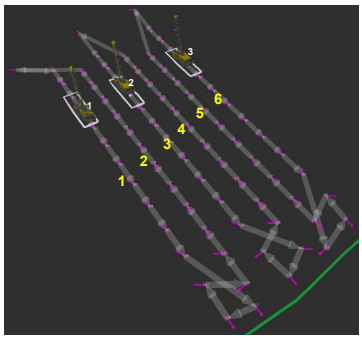

Figure 10. A solution to the DP^3 problem in this bench which is visualized in the ROS visualization tool, white arrows depict sequencing, pink arrows represent vehicle poses at each drill target, and the green line shows a part of the geofence.

machine 1 starts with row 1 and exits the field from row 2, machine 2 starts at row 3 and exits at row 4, and machine 3 starts its missions from row 6 and exits from row 5. The plan is found in 2.14 minutes.

Figure 11 shows three snapshots of a simulated run. Each snapshot shows the motions of each robot (the collection of all movement polygons). In Figure 11(a), the motions of robot 1 partly occupy rows 3 and 4 for manoeuvring between row 1 and 2. This results in robot 2 having to wait just before the conflicting area, until robot 1 finishes its manoeuvre. As robot 2 also occupies some part of rows 5 and 6 (see Figure 11(b)), robot 3 has to wait until robots 1 and 2 are finished switching rows.

The fact that the motions of machine 2 do not conflict with rows 1 and 2 is handled by the motion planner; the same holds for machine 3 and rows 3 and 4 (see Figure 11(c)). As the machines start their execution concurrently, their motions would lead to collisions, were it not for the fact that the coordination sub-problem was solved as well. The temporal constraints that were selected by the algorithm resolve these conflicts by forcing machines 2 and 3 to yield to machine 1.

The plan obtained with Algorithm `DP`^{3}`-solver` was run in simulation three more times. During the first run, we artificially injected delays in the drilling activities of robot 1 for a drill target in row 1. Through temporal reasoning, the delay was propagated on the start times of future missions of machines 2 and 3. The TTC increased drastically as a result, as robot 2 (and consequently robot 3) were forced to wait until robot 1 finished manoeuvring between rows 1 and 2. In the second run, we artificially delayed robot 1 while drilling a target in the second row. In this case, TTC only increased by the amount of the delay, since the delay does not affect robot 2 and robot 3. Finally, in the third run, we injected a delay in one of the initial drill targets of robots 2 and 3. Since these robots were scheduled to yield to robot 1 later on during execution, the delay did not increase the TTC. In all three runs, the contingencies were accounted for by

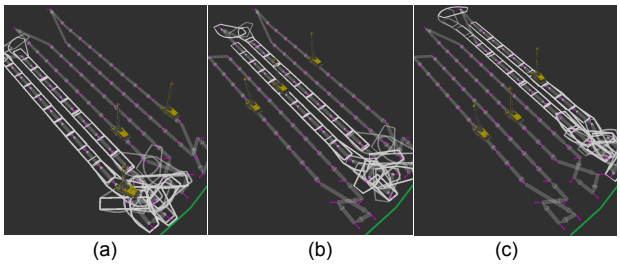

FIGURE 11. Motions of different robots represented on a separate figure as white convex polygons.

	27	53	76
h^{val}_{seq} w/o *sp*	8.68 min	timeout	timeout
h^{val}_{seq} w *sp*	0.21 min	0.6 min	2.14 min

TABLE 1. A small quantitative evaluation h^{val}_{seq}.

temporal reasoning and the resulting adjusted plan was dispatched to the machines within the 1 second sampling period.

We also evaluated the effectiveness of the sequence pattern (*sp*, see Section 5) used by h^{val}_{seq} on problems defined over the same drill pattern as was used in the previous experiments. We considered three cases with a different number of drill targets (27, 53 and 76). Each case was run twice, once with and once without the *sp* heuristic. The search process was aborted after a 60 minute timeout. Table 1 shows the strong pruning power of this heuristic.

In the next experiment, the focus was on evaluating the TTC in a slightly bigger setup with 91 drill targets (see Figure 12). The TTC of the solution, where machine 1 starts at row 1 and exits at row 4 and machine 2 starts at row 5 and exits at row 8, is 101 minutes, whereas the TTC is 61 min in the case that machine 1 starts at row 4 and exits at row 1 and machine 2 starts at row 5 and exits at row 8. In the latter case, the machines do not need to wait for each other and all the operations can be done in parallel. In order to extract the latter solution from the search space, an ad-hoc sequencing heuristic built for this particular set of drill targets was employed. This proves that this solution is in the search space. As has been mentioned, devising a general heuristic that biases the sequencing choices to minimize TTC is a topic for future work.

7.1. Comparison with A^*

Our approach consists in exploring the joint search space of different sub-problems via a heuristically informed CSP-style backtracking search. The intelligence in our approach is distributed among several heuristics, each guiding the resolution of specific sub-problems. An obvious alternative approach is to define the state of DP^3 as node in a traditional heuristic search, and to employ A* (or some other heuristic search algorithm) in conjunction with a heuristic that

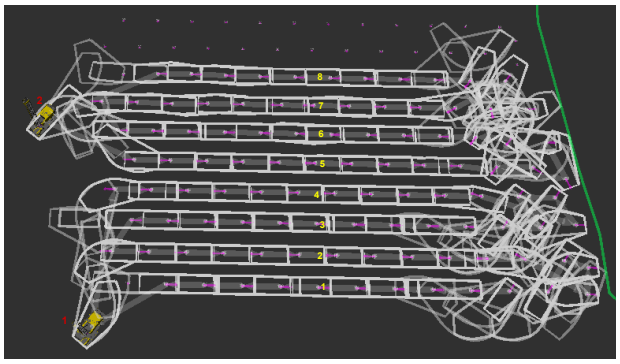

FIGURE 12. A high quality solution with respect to the TTC for a bench with 91 drill targets.

accounts for all aspects of the DP^3. We would, however, have to build all aspects of the problem into this single heuristic — that is, we would have to make the heuristic capable of informing which are the best allocations of vehicles to targets, the best sequencing, grouping and scheduling decisions, etc. More importantly, it would be difficult to exploit the structure of the different sub-problems to enhance performance. In our approach, scalability can be improved by identifying heuristics for specific sub-problems and/or heuristics that exploit the hybrid nature of the underlying representation. Another important factor in industrial domains is the ability to specify domain-dependent requirements, e.g., in this specific bench, that drill targets 122 and 344 must be drilled at the end, or that machine 3 should not drill a set of drill targets. The modularity of our approach facilitates programming these types of requirements in the relevant decision variables of specific sub-problems.

8. Related Work and Outlook

We have divided DP^3 into several sub-problems and the `DP3-solver` finds mutually feasible solutions to the individual sub-problems. The idea of problem decomposition and solution synthesis is not new in AI. There have been many studies in Multi Agent Systems (MAS) regarding cooperative problem solving [8]. Although our work has not been done in the context of MAS, we reify this idea into a challenging robotic domain.

The `DP3-solver` combines solutions from different sub-problems, each of which can be seen as a "classical" AI or robotics problem. The sequencing sub-problem can be seen as a Vehicle Routing Problem with Multiple Depots (MDVRP) where the drill targets are the customers. MDVRP is a combinatorial optimization which has been largely studied. However, solutions to MDVRPs often do not consider spatial, temporal and kinematic constraints, nor dynamic maps. The need to consider these problems destroys the assumptions based on which existing AI solutions are built. A comprehensive survey [9] has shown, that there are various ways of modeling for MDVRP including time

windows, split delivery, heterogeneous fleet according to the single and multiple objectives. For future work, our main focus is to optimize the TTC while all the requirements are upheld, therefore, casting the sequencing sub-problem to MDVRP is therefore an option for further investigation.

Combined route and motion planning in the presence of strong spatial and temporal constraints has been studied [10]. Although the application differs, the proposed approach is similar to ours in that it combines solutions from different sub-problems such as non-trivial motion planning and route planning. However, the solution assumes that sequencing of goal poses is given, and it does not handle on-line contingencies.

The coordination problem has to be addressed when we have a fleet of autonomous vehicles. Many approaches to this problem largely rely on fixed trajectories (e.g., [11] and the KIVA system [12, 13]). This makes the coordination problem much easier than in our case, where vehicle paths are not known *a priori*. The coordination sub-problem could be addressed by using multi-robot motion planners, using a distributed approach [14] or solving the problem in a centralized fashion [15]. The latter is similar to our work in terms of using a centralized approach. However, multi-robot motion planners are not efficient enough for use within another search, and are unable to handle temporal contingencies that may occurr on-line.

In this work, we have broken down a given hybrid problem and have identified interdependency among the sub-problems, and we interleave reasoning within each sub-problem. A heuristically guided backtracking search finds a solution to the overall problem in the joint search spaces of these sub-problems. This approach is general and can be used in other domains, such as task planning for mobile service robots [16] or warehouse management [17].

In addition to focusing on the optimization issue, our ongoing work will broaden the range of possible online contingencies (e.g., machine breakdowns which require removing a machine) that can be dealt with by our `DP`^{3}`-solver`.

Acknowledgements

This work has been supported by the Swedish Knowledge Foundation (KKS) project "Semantic Robots" and by Atlas Copco. We are grateful to Pontus Bergsten, Lars Eriksson, Stephen Joyce, Robert Lundh, and Ola Pettersson for their support. We also wish to thank Nils Johansson and the workers of the New Boliden mine in Gällivare for their precious insights into the mining process.

References

[1] H. Andreasson, A. Bouguerra, M. Cirillo, et al. Autonomous transport vehicles: where we are and what is missing. *Robotics & Automation Magazine, IEEE* **22**(1):64–75, 2015. doi:10.1109/MRA.2014.2381357.

[2] E. Tsang. *Foundations of constraint satisfaction.* Computation in cognitive science. Academic Press, 1993.

[3] T.-C. Liang, J.-S. Liu, G.-T. Hung, Y.-Z. Chang. Practical and flexible path planning for car-like mobile robot using maximal-curvature cubic spiral. *Robotics and Autonomous Systems* **52**(4):312–335, 2005. doi:10.1016/j.robot.2005.05.001.

[4] R. Dechter, I. Meiri, J. Pearl. Temporal constraint networks. *Artificial Intelligence* **49**(1-3):61–95, 1991. doi:10.1016/0004-3702(91)90006-6.

[5] A. Cesta, A. Oddi, S. F. Smith. A constraint-based method for project scheduling with time windows. *Journal of Heuristics* **8**(1):109–136, 2002. doi:10.1023/A:1013617802515.

[6] S. Qin, T. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice* **11**:733–764, 2003. doi:10.1016/S0967-0661(02)00186-7.

[7] M. Quigley, K. Conley, B. Gerkey, et al. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*. 2009.

[8] M. Woolridge. *Introduction to Multiagent Systems.* John Wiley & Sons, Inc., New York, NY, USA, 2001.

[9] J. R. Montoya-Torres, J. L. Franco, N. I. Santiago, et al. Survey. *Computers & Industrial Engineering* **79**(Complete):115–129, 2015. doi:10.1016/j.cie.2014.10.029.

[10] S. Scheuren, S. Stiene, R. Hartanto, et al. Spatio-temporally constrained planning for cooperative vehicles in a harvesting scenario. *KI* pp. 341–346, 2013. doi:10.1007/s13218-013-0267-y.

[11] J. Marshall, T. Barfoot, J. Larsson. Autonomous underground tramming for center-articulated vehicles. *Journal of Field Robotics* **25**(6-7):400–421, 2008. doi:10.1002/rob.20242.

[12] Amazon robotics. `http://www.amazonrobotics.com`. Accessed: 2015-09-01.

[13] E. Guizzo. Three engineers, hundreds of robots, one warehouse. *IEEE Spectrum* **45**(7):26–34, 2008.

[14] K. E. Bekris, D. K. Grady, M. Moll, L. E. Kavraki. Safe distributed motion coordination for second-order systems with different planning cycles. *Int Journal of Robotics Research (IJRR)* **31**(2), 2012. doi:10.1177/0278364911430420.

[15] M. Cirillo, T. Uras, S. Koenig. A lattice-based approach to multi-robot motion planning for non-holonomic vehicles. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2014. doi:10.1109/IROS.2014.6942566.

[16] S. Stock, M. Mansouri, F. Pecora, J. Hertzberg. Online task merging with a hierarchical hybrid task planner for mobile service robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015.

[17] M. Mansouri, H. Andreasson, F. Pecora. Towards hybrid reasoning for automated industrial fleet management. In *Proc. of the IJCAI Workshop on Hybrid Reasoning*. 2015.